**PDF Active-Reading Assistant  
Initial Software Design Specification**

Ryan Helms (rh), William Qiu (wq), Nikhar Ramlakhan (nr), Abie Safdie (as), Caleb Sutherland (cs)

4-12-2024 - v1.02

**Table of Contents**

[1. SDS Revision History 1](#_Toc164978254)

[2. System Overview 1](#_Toc164978255)

[3. Software Architecture 2](#_Toc164978256)

[4. Software Modules 3](#_Toc164978257)

[4.1. PDF Viewer 3](#_Toc164978258)

[4.2. Annotation System 4](#_Toc164978259)

[4.3. User Selection Interface 6](#_Toc164978260)

[4.4. Data Storage Server 7](#_Toc164978261)

[4.5. Self-Quizzing 10](#_Toc164978262)

[5. Dynamic Models of Operational Scenarios 12](#_Toc164978263)

[6. References 13](#_Toc164978264)

[7. Acknowledgements 13](#_Toc164978265)

# 1. SDS Revision History

**Date Author Description**

**————— ——— ————————————————————————————**

4-10-2024 nr Created the initial document.

4-11-2024 as, cs Provided models, descriptions, and general edits for Software Modules Section

4-12-2024 team Added static and dynamic module diagrams.

4-25-2024 cs Final revisions to SDS

# 2. System Overview

The system, named the PDF Active-Reading Assistant, aims to provide users with tools to enhance their active reading experience with PDF documents. It will consist of components for accessing and organizing PDF files, note-taking, as well as a server component for storing user data.

# 3. Software Architecture

The software architecture of the PDF Active-Reading Assistant is designed to facilitate efficient interaction between its various components, ensuring seamless functionality and ease of maintenance. The architecture is composed of several key components, each serving a specific role in the system.

1. Set of Components:

* PDF Viewer: Responsible for accessing PDF files and rendering their content for display to the user.
* Annotation System: Facilitates user annotation of PDF documents, providing tools for highlighting, commenting, and organizing annotations.
* User Selection Interface: Responsible for validating users and linking data to the correct user.
* Self-Quizzing: Responsible for testing knowledge acquired by applying the SQ3R method.
* Data Storage Server: Manages the storage and retrieval of user data, including annotations and user profiles.

2. Functionality Provided by Each Component:

* PDF Viewer: Renders PDF content on the user interface, allows navigation through the document, and interfaces with the Annotation System to display user annotations.
* Annotation System: Provides tools for creating, editing, and organizing user annotations, stores annotations in the server, and interfaces with the PDF Viewer for display.
* User Selection Interface: Acts as a homepage for all user, allowing them to each separately access their PDFs and associated notes.
* Self-Quizzing: Displaying and hiding note taking components.
* Data Storage Server: Stores user data, manages user authentication, and provides APIs for communication between client-side components.

3. Interaction Between Modules:

* The PDF Viewer interacts with the Annotation System to display user annotations alongside the PDF content, enabling users to view and manage their annotations in real-time.
* The Annotation System communicates with the Data Storage Server to store and retrieve user annotations, ensuring data persistence across sessions and devices.
* The User Selection Interface module interacts with the Data Storage Server module to link user data
* The Self-Quizzing interacts with the Annotation System to hide and display user notes.

4. Rationale for the Architectural Design:

The architectural design was chosen to promote modularity, scalability, and maintainability of the system. By decomposing the system into separate components, each responsible for a specific aspect of functionality, we ensure that changes or updates to one component do not affect the others. Additionally, the use of a server component allows for centralized data storage and management, facilitating collaboration and synchronization among multiple users. Overall, this architecture provides a solid foundation for the development of a robust and feature-rich PDF Active-Reading Assistant.

# 4. Software Modules

## 4.1. PDF Viewer

***Role and Primary Function***

The PDF Viewer module is responsible for rendering PDF files within the application's user interface. Its primary function is to display PDF content to users, allowing them to navigate through the document, zoom in/out, and interact with annotations.

***Interface to Other Modules***

* Interfaces with the Annotation System module to display user notes alongside the PDF content.
* Communicates with the application to provide tools for viewing the PDF. Such as loading, zooming, and scrolling.

***Static Model***

A diagram of a diagram

Description automatically generated

Figure 1

The static model (Figure 1) of the PDF Viewer module depicts its key components and their relationships. It includes components for loading a PDF, zooming, scrolling, and showing/hiding the notetaking area.

***Dynamic Model***

A diagram of a diagram

Description automatically generated

Figure 2

The dynamic model (Figure 2) illustrates the flow of control and data within the PDF Viewer module. For instance, if the user attempts to load a file that is invalid, it will be prompted to load another PDF. Otherwise, it will be granted access to the PDF viewing tools.

***Design rationale***

The PDF Viewer module was designed to provide users with a seamless and intuitive experience for viewing PDF documents within the application. Its separation from other modules allows for independent development and maintenance, ensuring flexibility and scalability. By incorporating both static and dynamic models, the design rationale emphasizes clarity and completeness in describing the module's functionality and interactions.

***Alternative designs***

Several alternative designs could be considered for the PDF Viewer module, including different rendering engines, user interface layouts, and interaction paradigms. Ultimately, the chosen design will be selected for its balance of performance, usability, and compatibility with other system components. Alternative approaches will be documented and evaluated to ensure that the final design met the project's requirements and constraints.

## 4.2. Annotation System

***Role and Primary Function***

The Annotation System module is responsible for managing user-generated annotations within the application. Its primary function is to allow users to create, edit, and delete annotations alongside PDF documents, providing a collaborative environment for document review and discussion.

***Interface to Other Modules***

* Interfaces with the PDF Viewer module to display annotations alongside the PDF content.
* Communicates with the Data Storage Server module to store and retrieve annotation data from the database.

***Static Model***

A diagram of a software system

Description automatically generated

Figure 3

The static model (Figure 3) of the Annotation System module outlines its components for managing annotations, including storage, rendering, and user interaction functionalities.

***Dynamic Model***

A diagram of a software system

Description automatically generated

Figure 4

The dynamic model (Figure 4) illustrates the flow of annotation creation, editing, and deletion within the Annotation System module. It demonstrates how user actions trigger updates to annotation data and presentation.

***Design rationale***

The design of the Annotation System module prioritizes flexibility and scalability to accommodate various types of annotations and user interactions. By separating annotation management from PDF rendering, the module allows for independent development and extensibility. The dynamic model reflects the module's responsiveness to user actions and its integration with other system components.

***Alternative designs***

Alternative designs for the Annotation System module will be explored, including different data structures for storing annotations, user interface layouts for annotation editing, and synchronization mechanisms for collaborative annotation. The chosen design will be selected based on its suitability for supporting real-time collaboration, efficient data storage, and seamless integration with the PDF Viewer and Data Storage Server modules.

## 4.3. User Selection Interface

***Role and Primary Function***

The User Selection Interface is responsible for verifying which user is requesting to access the application. Its primary function is to grant access and provide the database component of the software with the information needed to load the appropriate data stored on the server.

***Interface to Other Modules***

* Interfaces with the application to provide user identification.
* Communicates with the Data Storage Server module to store data with respect to each user.

***Static Model***

A diagram of a computer network

Description automatically generated

Figure 5

The static model (Figure 5) of the User Selection Interface outlines how the module interacts with the Data Storage Server.

***Dynamic Model***

A diagram of a login

Description automatically generated

Figure 6

The dynamic model (Figure 6) illustrates the User Selection Interface with respect to valid information. Upon selecting their profile, the application will load their PDFs and associated notes.

***Design rationale***

The design of the User Selection Interface focuses on ease-of-use and security. By having a simple username and password field, the user will be able to quickly input their credentials to access the site. Security is ensured by deploying access control. Meaning, if the user inputs valid credentials they will be given access to *only* their notes and other information that they stored on the server.

***Alternative designs***

Alternative designs for the User Selection Interface will be considered on different effectiveness and ease of use for the user. The intent is for the user to sign-in and access their respective data with ease.

## 4.4. Data Storage Server

***Role and Primary Function***

The Data Storage Server module is responsible for managing application data, including user accounts, annotations, and PDF document metadata. Its primary function is to provide a centralized repository for storing and retrieving data, ensuring data integrity, security, and scalability.

***Interface to Other Modules***

* Interfaces with the Annotation System module to store and retrieve annotation data.
* Communicates with the PDF Parser module to store parsed content metadata.
* Provides APIs for client modules to access and manipulate data stored in the database.

***Static Model***

A diagram of a computer

Description automatically generated

Figure 7

The static model (Figure 7) of the Data Storage Server module outlines its components for data management, including database management, authentication, and data access functionalities.

***Dynamic Model***

A diagram of a software application

Description automatically generated

Figure 8

The dynamic model (Figure 8) illustrates the flow of data within the Data Storage Server module, demonstrating how client requests are processed, authenticated, and served, and how data is retrieved from and stored in the database.

***Design rationale***

The design of the Data Storage Server module emphasizes reliability, security, and scalability in managing application data. By employing a client-server architecture, the module enables concurrent access from multiple clients while enforcing access controls and data consistency. The dynamic model reflects the module's handling of client requests and its integration with other system components for seamless data exchange.

***Alternative designs***

Alternative designs for the Data Storage Server module explored different database technologies, authentication mechanisms, and data storage models. The chosen design was selected based on its suitability for supporting concurrent access, data encryption, and backup/restore functionalities, while allowing for future enhancements and optimizations.

## 4.5. Self-Quizzing

***Role and Primary Function***

The Self-Quizzing module is responsible for allowing the user to quiz themselves with the knowledge they acquired from reading the PDF document in a SQ3R manner.

***Interface to Other Modules***

* Interfaces with the Annotation System module to show/hide notes that the user took.
* Interfaces with the Database to load notes that were inputted by the user.

***Static Model***

A diagram of a diagram

Description automatically generated

Figure 9

The static model of the Self-Quizzing module (Figure 9) showcases its interaction to different components of the application. These include the Notetaking area, the hide/show button, the database, and the PDF. As a result, the Self-Quizzing module will have the ability to show/hide notes, load/store notes to the database, and change the size and view of the PDF.

***Dynamic Model***

A diagram of a computer system

Description automatically generated

Figure 10

The dynamic model (Figure 10) illustrates the flow of control for the Self-Quizzing module. The user will be able to hide/show notes and load/store data from the database.

***Design rationale***

The design of the Self-Quizzing module will be focused on the showing/hiding of the notetaking area. At the forefront of design will be ease-of-use and reliability. The showing/hiding of notes will be enacted via a single button press. The reliability of notes loading/storing from the database will be necessary to establish.

***Alternative designs***

Alternative designs for the Self-Quizzing module will be considered as development progresses. If there is an optimization that improves ease-of-use and reliability it will be enacted.

# 5. Dynamic Models of Operational Scenarios

A diagram of a company

Description automatically generated

Figure 11

Figure 11 illustrates the primary use case scenario for PDF Active-Reading-Assistant. The diagram highlights that the initial step involves user authentication, followed by the ability to upload a PDF for note-taking or utilizing PDF tools. Regarding note-taking functionality, users can seamlessly display, load, and store notes within the database. As for PDF tools, users have the capability to zoom in and scroll through the document effortlessly.

# 6. References

Faulk, Stuart. (2011-2017). CIS 422 Document Template. Downloaded from https:// uocis.assembla.com/spaces/cis-f17-template/wiki in 2018. It appears as if some of the material in this document was written by Michal Young.

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design— Software Design Descriptions. <https://ieeexplore.ieee.org/document/5167255>

J, Anthony Hornof. “Software Design Specification [Template].” 6 May 2019.

Sommerville, I. (2000) Software Engineering. Addison-Wesley, Harlow.

# 7. Acknowledgements

This document is built with reference to the SDS template provided by Prof. Anthony Hornof. Additionally, it builds on a document developed by Stuart Faulk in 2017, and on the publications cited within the document, such as IEEE Std 1016-2009.